Toteutusdokumentti

Ohjelma koostuu yhdeksästä luokasta: App on sen käynnistävä luokka, jossa luodaan muut luokat ja annetaan alkupisteen ja loppupisteen arvot, sekä valitaan mitä kolmesta algoritmista halutaan sillä kertaa ajaa. Piste-luokka sisältää koordinaatistopisteet, joita algoritmi käyttää pohjanaan.

AstarAlgoritmi-luokka sisältää itse haku-algoritmin, joka toimii kuten yleisesti tunnettu A*. Se tuntee myös luokan AstarVertailija, jota se käyttää yhdessä keon kanssa antamaan kulloinkin lähimmän pisteen maaliin nähden.

Djikstran algoritmi on hyvin samanlainen kuin astar, mutta hieman hitaampi. Toisaalta se löytää varmemmin optimaalisen reitin. Usein käytetään kuitenkin astari sen ollessa niin paljon nopeampi eikä tuloskaan ole kaukana optimaalisesta.

BFS on leveyssuuntainen haku kahden pisteen valilla. Se on hyvin hidas verrattuna Djikstraan ja Astariin.

Etäisyys on laskettu linnuntietä, mutta pisteiden välillä voi liikkua vain joko ylös, alas, vasemmalle tai oikealle. Keko on itse toteutettu tietorakenne, joka jäljittelee javan PriorityQuen toimintaa. Se tarvitsee toimiakseen sisälleen listan, joka myös on tässä itse toteutettu. Lista on tehty toimimaan samalla tavalla kuin javan ArrayList toimisi vastaavassa tilanteessa.

Huonoimman tapauksen aikavaativuus astarille on O(|V| + |E|) log |V|) ja huonoimman tapauksen tilavaatimus on O(|V|). Djikstralla aikavaatimukset ovat huonoimmassa tapauksessa samat kuin astarilla. Keskimääräinen haku-aika on kuitenkin ratkaiseva. Tässä E on polkujen määrä ja V solmujen määrä. Vastaavat BFS:lle ovat O(E + V) ja tilaa O(V) eli solmujen määrän verran.

Työtä vois parannella esimerkiksi muuttamalla pisteiden välistä liikkumista monimutkaisemmaksi. Esimerkiksi sallimalla liikkeet myös väli-ilmansuuntiin ja luomalla esteitä koordinaatistoon. Myös tietorakenteita voisi parannella niin, että ne toimisivat useammissa tilanteissa. Eli lisätä metodeja, joita on esim javan ArrayListilläkin.

Lähteet:

http://bigocheatsheet.com/
http://en.wikipedia.org/wiki/A*_search_algorithm
http://en.wikipedia.org/wiki/Breadth-first_search
http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
http://www.cs.helsinki.fi/node/81607
Tietorakenteet ja algoritmit materiaal - Patrik Florell
www.github.com – aikaisemmat työt.